A DISTRIBUTED PROGNOSTIC HEALTH MANAGEMENT ARCHITECTURE

Bhaskar Saha*, Sankalita Saha* and Kai Goebel#
*Mission Critical Technologies (NASA Ames Research Center)
#NASA Ames Research Center
Moffett Field, CA 94035
{bhaskar.saha, sankalita.saha-1, kai.goebel}@nasa.gov

Abstract: This paper introduces a generic distributed prognostic health management (PHM) architecture with specific application to the electrical power systems domain. Current state-of-the-art PHM systems are mostly centralized in nature, where all the processing is reliant on a single processor. This can lead to loss of functionality in case of a crash of the central processor or monitor. Furthermore, with increases in the volume of sensor data as well as the complexity of algorithms, traditional centralized systems become unsuitable for successful deployment, and efficient distributed architectures are required. A distributed architecture though, is not effective unless there is an algorithmic framework to take advantage of its unique abilities. The health management paradigm envisaged here incorporates a heterogeneous set of system components monitored by a varied suite of sensors and a particle filtering (PF) framework that has the power and the flexibility to adapt to the different diagnostic and prognostic needs. Both the diagnostic and prognostic tasks are formulated as a particle filtering problem in order to explicitly represent and manage uncertainties; however, typically the complexity of the prognostic routine is higher than the computational power of one computational element (*CE*). Individual *CE*s run diagnostic routines until the system variable being monitored crosses beyond a nominal threshold, upon which it coordinates with other networked *CE*s to run the prognostic routine in a distributed fashion. Implementation results from a network of distributed embedded devices monitoring a prototypical aircraft electrical power system are presented, where the *CE*s are Sun Microsystems Small Programmable Object Technology (SPOT) devices.

Key Words: Distributed system health management architecture; diagnostics; prognostics; embedded systems; remaining useful life; particle filters; distributed resampling.

I. INTRODUCTION

Distributed diagnostics and prognostics is the next step in the evolution of health management systems as expectations of health update frequency, system coverage and prediction accuracy increase. The most common architecture for health management systems – due to ease of development – is centralized i.e., a central processing device collects data from sensors and processes them, while executing various diagnostic and prognostic algorithms. However, such a system architecture, where all or most of the processing is reliant on a single processor, is prone to various problems; the most serious being vulnerability to complete loss of functionality in case of a crash of the central computing element. Furthermore, with increase in amount of sensor data as well as the complexity of algorithms, traditional centralized systems become unsuitable for successful deployment and efficient distributed architectures are required.

A distributed architecture though, is not effective unless there is an algorithmic framework to take advantage of its unique abilities. The health management paradigm envisaged here incorporates a heterogeneous set of system components monitored by a varied suite of sensors and a particle filtering (PF) framework that has the power and the flexibility to adapt to the different diagnostic and prognostic needs. The application domain for this research is an experimental setup simulating a prototypical aircraft electrical power system. The system components under test include power electronic devices and lithium-ion batteries. Sun Microsystems Small Programmable Object Technology (SPOT) devices are used for embedded sensing as well as computing. Each system component is monitored by a single Sun SPOT device that takes voltage, current and temperature measurements as appropriate and runs the diagnostic routine on the collected data. Once a fault is detected the SPOT device in question switches over to the prognostic routine where it enlists the help of other available SPOT devices to set up a distributed wireless network in order to share the prognostic computational load.

Both the diagnostic and prognostic tasks are formulated as a particle filtering problem in order to explicitly represent and manage the uncertainties inherent to the PHM domain. Such uncertainties could result from various sources like insufficient system model fidelity, sensor noise or unanticipated operating conditions. The diagnostic system model is a simple low order formulation that allows the limited computation power of a single SPOT device to track the measured variable indicative of component health and trigger the prognostic routine when certain predetermined thresholds are crossed. The prognostic model is more complex in order to handle the increased uncertainty of health prediction and hence load sharing over the distributed network is required to obtain real-time results. The cooperating SPOT devices multi-task to carry out individual diagnostic duties along with distributed prognostic tasks and maximize the overall health management system efficiency. Thus, the overall system involves complex run-time scheduling and decision-making. The results discuss the prediction accuracy and precision of the prognostic routine and the computational performance gains of the distributed architecture over classical centralized schemes.

II. BACKGROUND

Research into systems health management has traditionally been spearheaded by the manufacturing, power generation and military systems community. Rotating machinery were the first implementation platforms for these technologies, since downtime or breakdown incurred significant costs, both in terms of economic viability and resource availability. Forays into the world of electrical and electronic devices are more recent. Even so, most of the body of work available addresses the issue of diagnostics – fault detection, identification and isolation. The field of prognostics is still very much nascent, although with the advent of the Condition-based Maintenance (CBM) paradigm, the emphasis on failure prediction and prevention has increased.

Early efforts in machinery diagnostics were mostly data-driven techniques applied to vibration data [1], [2]. Integrated diagnostics and prognostics approaches have emerged in recent years. In [3] the authors discuss such a methodology using statistical prognostic models. They also emphasize the need for a physics-of-failure based approach to improve the reliability of prognostics. A reasoning engine for distributed multi-algorithm diagnostics and prognostics is presented in [4]. Distributed data mining tools like DAME [5] and BROADEN [6] have also been developed for aircraft engine health management applications.

Looking at the health management problem from the systems perspective, the authors in [7] present a distributed prognostics architecture where tasks, like identifying the different system modules and determining where they fit into a given system using prognostics, are distributed at the algorithm level. A distributed network of smart sensor elements integrated using a knowledge-driven environment is presented in [8], which participates in a hierarchy of health determination at sensor, process, and system levels. A hardware multi-cellular sensing and communication network (a smart "skin") for health management of "ageless" aerospace vehicles (AAVs) is presented in [9]. The objective is to detect and self-heal from impacts caused by projectiles like micro-meteoroids or space debris.

From the algorithms perspective, some prognostic techniques – such as particle filters – have been investigated from a distributed implementation context. Three different distributed implementations for particle filtering are presented in [10]. A parallel particle filter implementation on a shared-memory multiprocessor cluster is discussed in [11]. The issue of resampling in distributed particle filter architectures has been discussed in [12]. Distributed particle filters for sensor networks [13] and tracking applications [14] have also been explored in recent times. Communication issues are most often the highest contributor to resource management costs for generic distributed networks. Specifically, for wireless networks communication overhead can be higher by orders of magnitude as compared to other factors. As a mitigation strategy, an approximate dynamic programming approach that integrates the value of information and the cost of transmitting data over a rolling time horizon is presented in [15]. However, the above technique is specific to the context of object tracking with a distributed sensor network and may not be easily extended to other domains. In [16] the authors investigate the role of network topology in improving communication overheads. The problem of minimizing communication in general distributed systems is considered in a discrete-event formalism in [17], where the system is modeled as a finite-state automaton.

From the literature review presented above, it can be inferred that there are several challenges to designing an efficient distributed health management architecture. It must be flexible enough to be able to monitor a variety of subsystems using heterogeneous implementation platforms while balancing the trade-offs among computational performance, resource requirements and communication overheads. The following sections describe a distributed PHM architecture that addresses these issues while using particle filters as the underlying algorithm. A subset of this work concerning only the distributed prognostics aspect was presented at the International Conference on Prognostics and Health Management 2008 (PHM 2008) [18].

III. DISTRIBUTED PHM ARCHITECTURE

At the heart of the distributed PHM architecture is a network of smart sensor devices. These devices monitor the health of various subsystems or modules i.e., they perform diagnostics operations and trigger prognostics operations based on user defined thresholds and rules. An example of such a distributed prognostics system is shown in Figure 1. The sensor devices which we call computing elements (*CE*s) consist of a sensor or a set of sensors and a communication device i.e., a wireless transreceiver or wired communication capabilities besides an embedded processing element. In this paper we focus on wirelessly connected devices for enhanced flexibility. However, for many systems, wired connections may be preferred in order to overcome communication overheads associated with wireless systems.

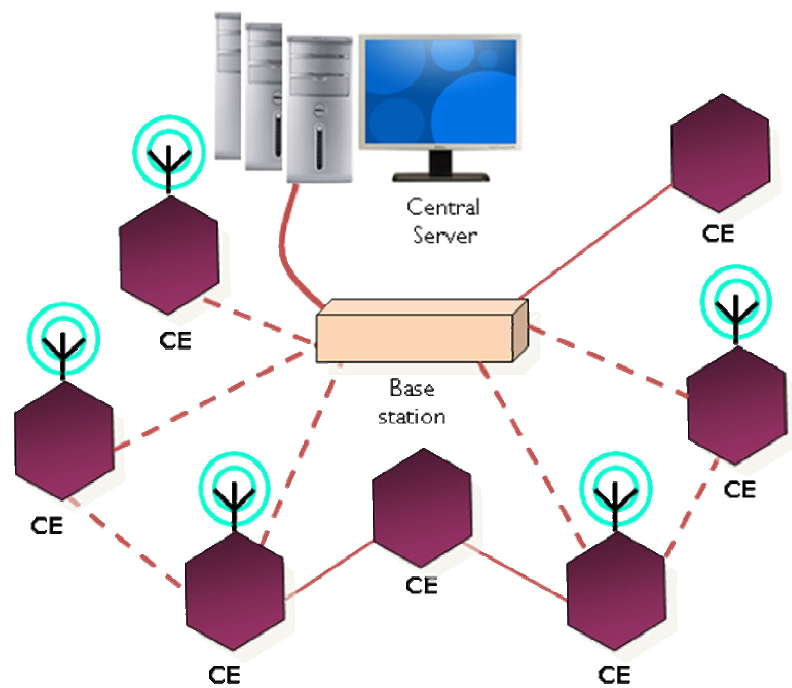


Figure 1 – Overview of distributed prognostics system architecture. Note that all the CEs may not have wireless connectivity. (Adapted from Figure 1 in [18]).

There are two main operating modes for a *CE*: *diagnostics* and *prognostics* [18]. A *CE* runs in the default mode of diagnostics where it monitors a given sub-system or component through a low weight diagnostic algorithm. During this monitoring if a *CE* detects a critical condition, it raises a flag. Depending on the current state (i.e., availability of resources) it either switches to prognostics mode or informs the base station of the prognostic task. Thus, in the prognostics mode it is not necessary that all *CE*s collaborate; some of them may lack enough computing power to support the additional new task. Note that the diagnostics operations continue in the prognostics mode. To ensure that a participating *CE* can support such multi-tasking efficiently the prognostics algorithms need to be distributed efficiently.

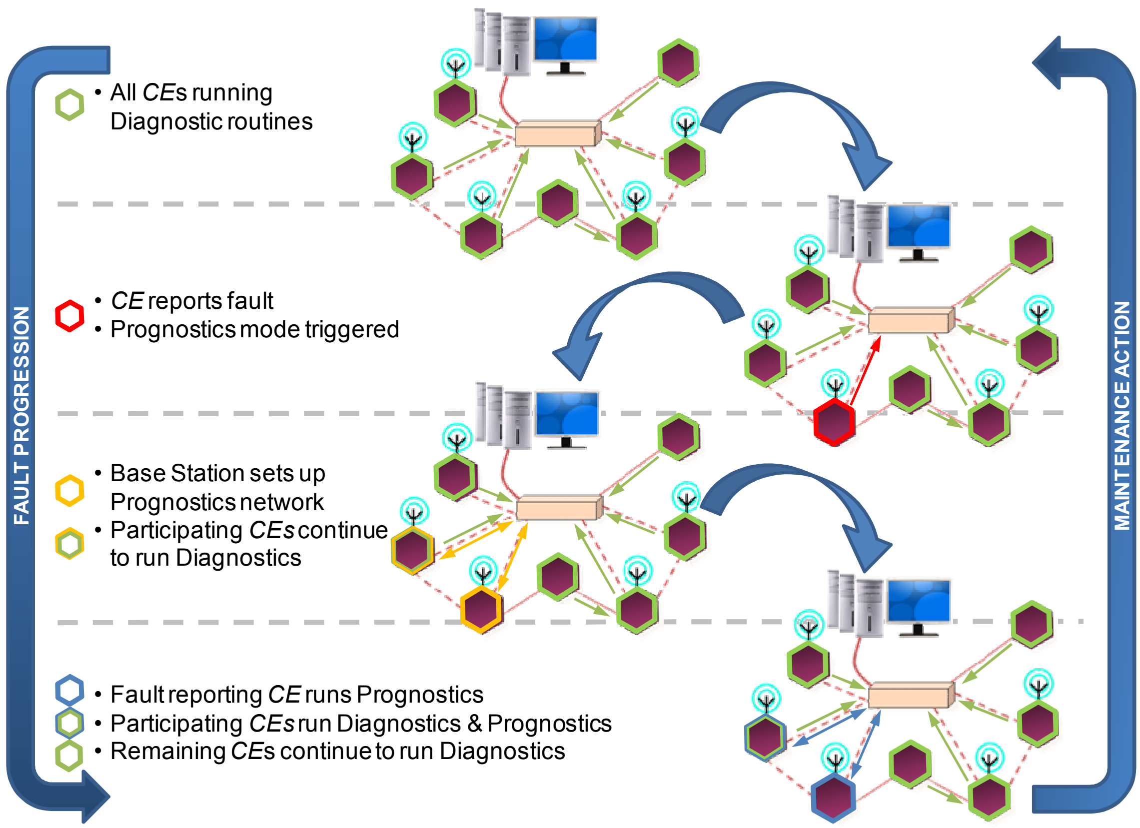


Figure 2 – Flow diagram for diagnostics and prognostics operations in the distributed architecture.

In many cases the sensor capabilities of the *CE*s may not be utilized, i.e., they could act as monitors for the rest of the system − schedule tasks, detect failures and initiate recovery, provide access to resources such as an external database etc. These *CE*s are specially designated as *base stations*. The *base station* is also, typically, connected to a more power computing resource such as a laptop which aids in collection and storage of system data.

Figure 2 shows, in detail, the typical execution flow in our health management architecture. As mentioned earlier, each *CE* monitors different components or subsystems such as battery health, actuator faults, health of electronic components and so on. It can also be responsible for diagnostic monitoring of a sub-system comprising of multiple components. In most cases the raw data collected is refined using diagnostics algorithms and only a summary is reported to the base station. But, in many cases, when the *CE* does not have enough computing power – for example in order to support heavy sampling rate of data collection – it can periodically send packets of raw data which may be used for offline analysis. Such a case is illustrated in our implemented architecture.

The *base station* monitors all the remaining *CE*s and coordinates tasks. The *base station* also maintains information regarding *CE* resource availability. Note that during prognostics, it is not necessary that the *base station* will coordinate all the tasks and another local leader/server may be chosen. The *CE*s involved in the prognostics operation would perform more efficiently if they are physically near to each other in a wireless environment and hence it is imperative that the *base station* is physically near to the rest of the *CE*s as well. Thus, in case the *base station* is far away from the collaborating *CE*s, a different leader is selected.

Implemented Architecture: In the system considered in this paper, two free ranging *CE*s (CE_1 and CE_2) are involved in addition to the *base station*, which also performs diagnostics on battery health data from an offline source. This scenario reflects the case when a *base station* has to aid some other *CE* in computation and illustrates the heavy multi-tasking involved in such a health management architecture. After startup of the system, during initialization, the *base station* communicates with remote *CE*s to gather information regarding available resources. Individual diagnostics routines are initiated by all the devices: CE_1 monitors temperature of an IGBT, CE_2 monitors temperature of a set of Lithium ion batteries and the *base station* runs offline diagnostics for battery based on current and voltage information. Further details of the diagnostic system are provided in Section V.

In our system, prognostics is triggered by the *base station* after it detects an anomaly. Based on resource information, it selects one of the *CE*s (say CE_1) to collaborate in prognostics on battery health data. It allocates task share to this *CE* and starts as well as acts as a leader for the prognostics routine. CE_1 now performs the prognostics sub-task in addition to its diagnostics task. The remaining free ranging *CE* (CE_2) continues its diagnostic operation. The *base station* now performs only its share of the prognostics task and scheduling and oversight of the prognostics task besides collection of diagnostics data from the two *CE*s. Once the required maintenance has been performed and the prognostics task is over the *base station* informs CE_1, which then returns to its diagnostics mode.

The Implementation Platform: The basic computational element of our implementation platform is the Sun Microsystems SPOT device. A free range Sun SPOT is a small, wireless, battery powered experimental platform built by stacking a Sun SPOT processor board with a sensor board and battery as shown in Figure 3. The smaller base station Sun SPOT consists of just the processor board in a plastic housing. In terms of processing power, each Sun SPOT has a 180MHz 32-bit ARM920T core processor with 512K RAM and 4M Flash. The Sun SPOTs communicate using radio channels. The processor board has a 2.4GHz radio with an integrated antenna on the board. The radio is a TI CC2420 (formerly ChipCon) and is IEEE 802.15.4 compliant. Each processor board has a USB interface (used to connect to a PC). Each free ranging SPOT runs off a 3.7 V rechargeable, 750 mAh Lithium-ion battery, while power management is carried out using a Atmel Atmega88 microcontroller. The base station SPOT does not have a battery, instead drawing power via the USB connection to the host PC. In our implementation, the free ranging SPOT act as *CE* while the SPOT base station acts as the default *base station*.

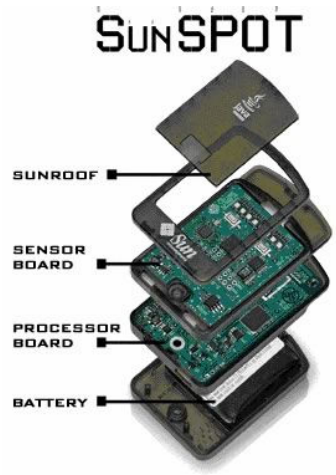


Figure 3 – Anatomy of a free ranging Sun SPOT device (courtesy of www.sunspotworld.com).

IV. PARTICLE FILTERS

In terms of the software program running on the above described architecture, we focus on a single-class of algorithms – particle filters. PF methods [19] are essentially Bayesian learning schemes that model the state equations as a first order Markov process with the outputs being conditionally independent. This has the advantage of making the next state prediction dependent only on the current state and the current measurement, which translates to lower memory and communication requirements than a Monte Carlo approach. PF methods are capable of identifying model parameters simultaneously with state estimation, thus tuning the system model to fault progression, making it superior to Kalman filters for health management approaches.

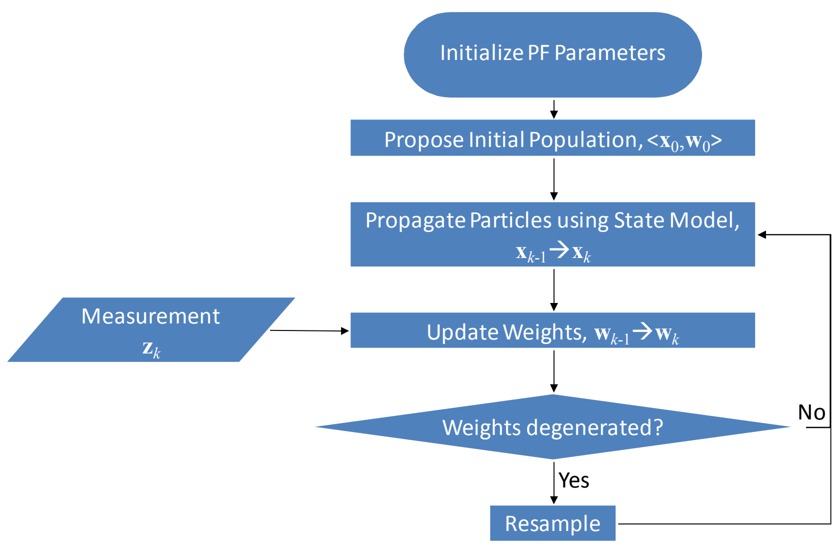


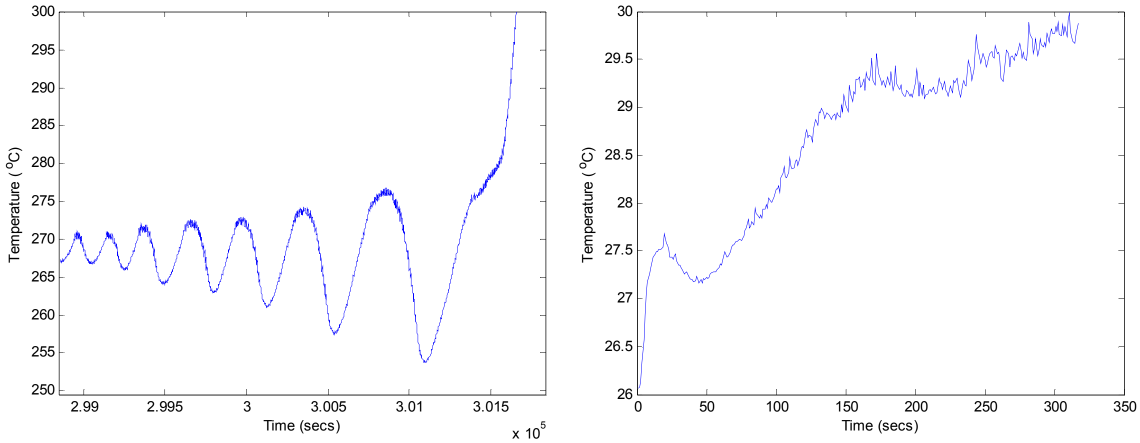
Figure 4 – Particle filter flow chart.

PFs approximates the state probability distribution (pdf) with a set of particles, $\mathbf{x}_k$, representing sampled values from the unknown state space, and their associated weights, $\mathbf{w}_k$, denoting discrete probability masses. Figure 4 shows a simplified flow chart for the PF algorithm. The particles are generated and recursively updated in two steps for each iteration of the filter – an intermediate estimate is generated from a stochastic nonlinear process model that describes the evolution in time of the system under analysis, which is then updated using the likelihood of the current measurement, $\mathbf{z}_k$, to produce the *posterior* state pdf. As the filter iterates and gets closer to the true state value, most of the particles weight degenerate, i.e. become negligible. Then, in order to prevent wasting computational resources on unimportant particles, resampling is carried out to generate a new population of particles concentrated around the more important ones of the old population. Details about this methodology are provided in [18].

V. DIAGNOSTICS

The diagnostic routine can be broken down into two parts – sensing and data analysis. For the prototypical aircraft electrical power system (EPS) under consideration, we look at a set of heterogeneous components, namely power semiconductor devices and batteries. Power semiconductors like Insulated-Gate Bipolar Transistor (IGBT) form the core of most electric power systems because of their high efficiency and fast switching speeds. However, due to high thermal and electrical stresses IGBTs undergo accelerated aging as compared to other electronic components that are part of the electrical power system. On aircrafts, batteries are mostly used for starting the engines and supply back up power to electrical loads (e.g. landing gear actuation, air conditioning etc.), while being charged as the engines run. However, it is still important to know the state-of-charge (SOC) and state-of-life (SOL) of the batteries for reliable operation, since their failure could impair operation of crucial systems leading to catastrophic consequences. Moreover, with the increasing role of electric UAVs in the future, batteries are going to be even more critical to overall system functionality.

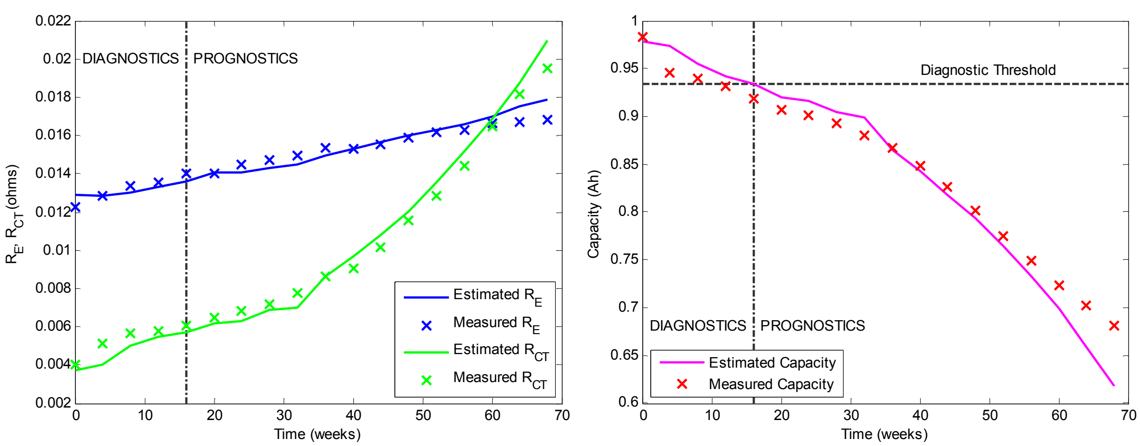
As mentioned before our set up consists of 2 SPOTs connected over a wireless network to a *base station* that is linked to the host PC. One of these SPOTs monitors the temperature of an IGBT as it undergoes repeated switching at elevated temperatures until loss of functionality due to thermal runaway or latch-up. The other SPOT looks at the temperature of a Lithium-ion rechargeable battery as it is repeatedly charged and discharged until it reaches a predefined end-of-life (EOL) capacity threshold (30% capacity fade). Figure 5 shows the temperature time series data collected by the SPOTs from both the IGBT and the Lithium-ion battery. Since run-to-failure experiments with these EPS components take considerable amount of time (more than the battery life of the free ranging SPOTs), pre-recorded battery aging data is fed to the monitoring SPOTs through the *base station* in order to show proof of concept of the integrated PF-based diagnostics-prognostics framework. The state variables of interest are the battery capacity, the electrolyte resistance (R_E) and the charge transfer resistance (R_{CT}) [18].



(a) (b)

Figure 5 – Temperature data collected from EPS components (a) IGBT, and (b) battery.

For diagnostics on offline battery aging data, the *base station* runs a lightweight version of the particle filter with only 20 particles, so as to fit within the computational and memory resources of a single *CE*. Using a low number of particles somewhat diminishes the ability of the PF to handle uncertainties, but since diagnostics is only concerned with tracking performance (1 step ahead prediction), the PF output is acceptable, as shown in Figure 6 (a) and (b). The tracking error for the internal impedance variables does not exceed 1milliohm while that for capacity drops to 10 mAh within 3 iterations. The diagnostic routine is run until the battery capacity as estimated by the *base station* crosses the 5% fade threshold, shown in Figure 6 (b), after which the prognostic routine is triggered. The state estimates at this trigger point serve as the starting population for the prognostics PF algorithm.



(a) (b)

Figure 6 – PF state tracking for diagnostics and state prediction during prognostics from 32 weeks onward for - (a) internal impedances R_E, R_{CT}, and (b) battery capacity.

VI. PROGNOSTICS

For the prognostic routine we need to make long-term predictions over several weeks and hence, we need more particles in the PF implementation to manage the uncertainty bounds. However, when the number of particles is stepped up to 100, the computational and memory resources available in a single *CE* is insufficient to handle the load. The *base station* then communicates with one of the *CEs* requesting help to set up a distributed computational network. These 2 nodes then execute the prognostic PF while working with 50 particles each. The distributed architecture takes advantage of the fact that there are no data dependencies during model-based particle propagation and the weight updates. These portions of the PF algorithm as easily parallelized, while only the resampling part is serial in nature. In our setup, the base station performs the resampling and particle routing, as well as overall control. More details about the execution steps can be found in [18].

Figure 6 (a) and (b) show the predicted trajectories of R_E, R_{CT} and battery capacity from 32 weeks onward. Although, by this point we have seen only half the life of the battery, the predictions are fairly accurate due to the ability of the PF to adapt the system aging model during the diagnostic routine. Furthermore, the PF does not simply provide the mean prediction trajectories, but also the predicted state pdf. This distribution may be compared against the EOL threshold (30% capacity fade, i.e. a battery capacity of 0.7 Ah) to generate the remaining useful life (RUL) pdf. Figure 7 shows how the PF prognosis improves in both accuracy and precision (narrowness of the pdf) from 32 weeks to 48 weeks as more data is made available before prediction.

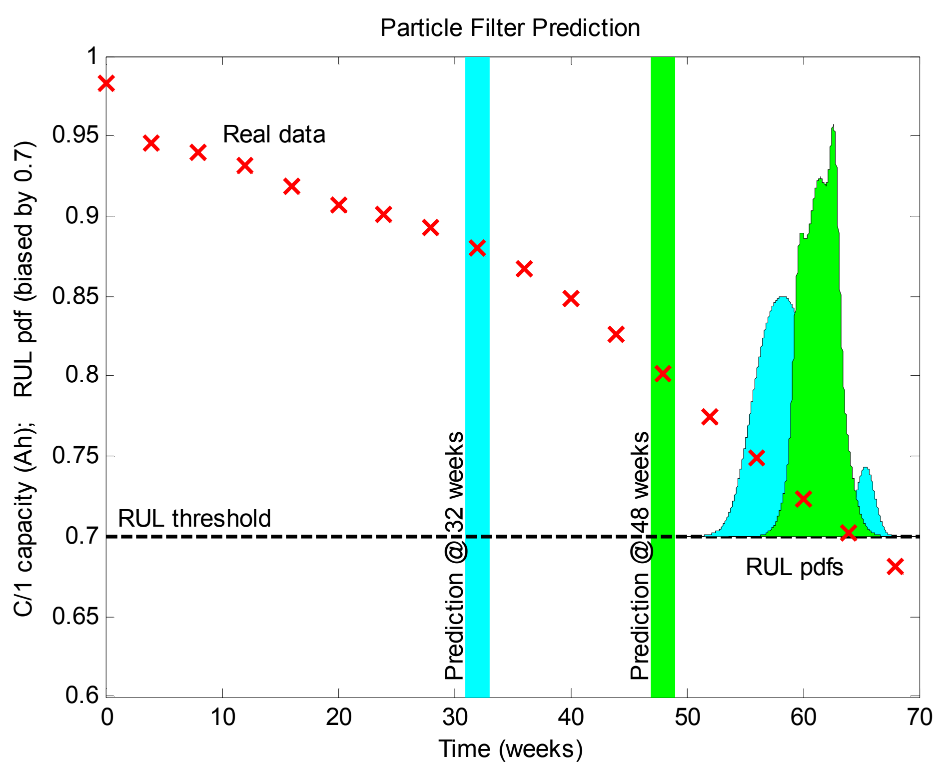


Figure 7 – Distributed PF prediction at 32 and 48 weeks.

VII. COMPUTATIONAL PERFORMANCE

The execution time profiles (averaged over multiple separate executions of the whole system) for prediction after 32 weeks and prediction after 48 weeks are shown in Figure 8. An average was taken since the execution time varies – within a margin of 10-15 ms – mainly based on the wireless communication time which is dependent on the distance between the *CE*s. The comparison shows the significant decrease in execution time when two *CE*s are used for prognostics, which is expected since the computation intensive load is now distributed. The inclusion of diagnostics task adds a minor overhead to the performance.

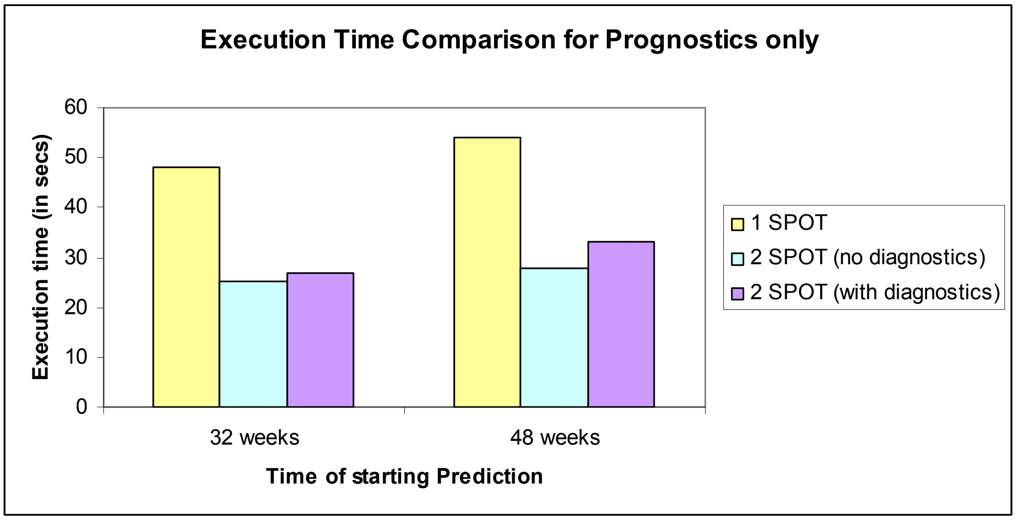


Figure 8 – Execution time comparison for health management system.

The memory usage of the CE_1, CE_2 and the base station are 29 KB, 28 KB and 25 KB respectively. Note that the *CE*s i.e., remote SPOTs execute using a virtual machine which contributes to additional memory use. The static program memory usage of one of the SPOT devices is illustrated in Figure 9. The memory usage reflects that most of the application memory is unused and further multi-tasking is possible thereby facilitating the scope for design of more complex health management system.

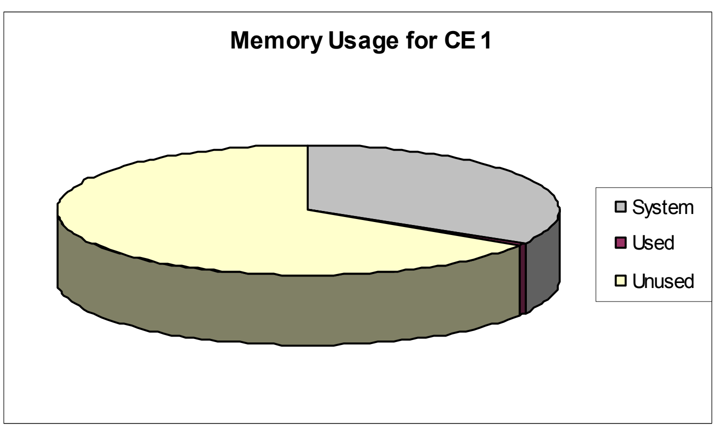


Figure 9 – Typical memory usage profile for a SPOT device.

VIII. CONCLUSION

This paper presents a distributed system health management architecture that handles diagnostics as well as prognostics operations in a collaborative manner. It builds on recent advances in smart sensor devices to distribute macro and micro level tasks, for better performance and resource utilization. The distributed prognostics part was highlighted in [18]. In this paper, a proof-of-concept demonstration of the architecture with the diagnostics workload thoroughly integrated has been presented which resulted in a heavily multi-tasking system. Experiments with monitoring a small heterogeneous set of EPS components to evaluate the effects on performance and resource (memory) utilization have been presented. Analysis of the results show that such a distributed architecture results in a highly efficient system (low execution time and static memory usage) that is capable of supporting complex functionalities like PF based diagnostics and prognostics, while providing high throughput rate.

The main focus of this paper has been architectural issues. However, new algorithmic modifications are also required to scale this system for real-time practical applications. Such algorithm explorations, besides investigation of more complex systems, are directions for future work. More analysis of architectural features such as communication protocols, power management, sophisticated partitioning and scheduling of tasks etc. are also important future research objectives.

REFERENCES

[1] R. H. Lyons, Machinery Noise and Diagnostics, Butterworth-Heinemann, 1987.

[2] M. F. Dimentberg, K. V. Frolov, and A. I. Menyailov, Vibroacoustical Diagnostics for Machines and Structures, Research Studies Press Ltd., 1991.

[3] Kam W. Ng, "Integrated Diagnostics and Prognostics of Rotating Machinery", International Journal of Rotating Machinery, vol. 5, no. 1, pp. 35-40, 1999.

[4] B. H. Bennett, P. Bergstrom, G.D. Hadden, G. J. Vachtsevanos, and J. Van Dyke, "Distributed Multi-Algorithm Diagnostics and Prognostics for US Navy Ships", 2002 AAAI Spring Symposium, 2002.

[5] T. Jackson, J. Austin, M. Fletcher, and M. Jessop, "Delivering a Grid enabled Distributed Aircraft Maintenance Environment (DAME)", in Proceedings of EPSRC AHM, Nottingham 2003.

[6] M. Fletcher, T. Jackson, M. Jessop, S. Klinger, B. Liang, J Austin, "The BROADEN Distributed Tool, Service and Data Architecture", in Proceedings of EPSRC AHM, Nottingham 2006.

[7] M. Roemer, C. Byington, G. Kacprzynski and G. Vachtsevanos, "An Overview of Selected Prognostic Technologies with Reference to an Integrated PHM Architecture", in Proc. of the First Intl. Forum on Integrated System Health Engineering and Management in Aerospace, 2005.

[8] J. Schmalzel, F. Figueroa, J. Morris, S. Mandayam and R. Polikar, “An Architecture for Intelligent Systems Based on Smart Sensors”, IEEE Transactions on Instrumentation and Measurement, vol. 54, no. 4X, pp. 1612-1616, Aug 2007.

[9] M. Prokopenko, P. Wang, D. C. Price, P. Valencia, M. Foreman, A. J. Farmer, “Self-organizing Hierarchies in Sensor and Communication Networks”. Artificial Life, Special Issue on Dynamic Hierarchies, vol. 11, no. 4, pp. 407-426, 2005.

[10] A. S. Bashi, V. P. Jilkov, X. R. Li and H. Chen, “Distributed Implementations of Particle Filters”, in Proc. of Sixth International Conference of Information Fusion, 2003, vol. 2, pp. 1164- 1171, 2003.

[11] S. Saha, C. Shen, C. Hsu, A. Veeraraghavan, G. Aggarwal, A. Sussman and S. S. Bhattacharyya, “Model-based OpenMP Implementation of a 3D Facial Pose Tracking System”, in Proc. of the Workshop on Parallel and Distributed Multimedia, Columbus, Ohio, pp. 66-73, Aug 2006.

[12] M. Bolic, P. M. Djuric and S. Hong, “Resampling Algorithms and Architectures for Distributed Particle Filters”, IEEE Transactions on Signal Processing, vol. 53, issue 7, pp. 2442- 2450, July 2005.

[13] M. Rosencrantz, G. Gordon and S. Thrun, “Decentralized Sensor Fusion with Distributed Particle Filters”, in Proc. Conf. Uncertainty in Artificial Intelligence Acapulco, Mexico, Aug 2003.

[14] X. Sheng, Y.-H. Hu and P. Ramanathan, “Distributed Particle Filter with GMM Approximation for Multiple Targets Localization and Tracking in Wireless Sensor Network”, in Fourth Intl. Symp. on Information Processing in Sensor Networks, pp. 181- 188, 2005.

[15] J. L. Williams, J. W. Fisher and A. S. Willsky, “Approximate Dynamic Programming for Communication-Constrained Sensor Network Management”, IEEE Transactions on Signal Processing, vol. 55, issue 8, pp. 4300-4311, Aug 2007.

[16] S. S. Iyengar, Q. Wu, and N. S. V. Rao, “Networking paradigm for distributed sensor networks”, in Proc. of the Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, pp. 284-290, Sept. 8-10, 2003.

[17] W. Wang, S. Lafortune and F. Lin, “A Polynomial Algorithm for Minimizing Communication in a Distributed Discrete Event System with a Central Station”, in Proc. Of 45th IEEE Conference on Decision & Control, San Diego, CA, USA, December 13-15, 2006.

[18] S. Saha, B. Saha, and K. Goebel, “Distributed Prognostics Using Wireless Embedded Devices”, in Proc. of International Conference on Prognostics and Health Management 2008 (PHM 2008), Oct 2008.

[19] S. Arulampalam, S. Maskell, N. J. Gordon and T. Clapp, “A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking”, IEEE Trans. on Signal Processing, vol. 50, no. 2, pp. 174-188, 2002.